# Java'da İşleçler, Kontrol Akış ve Döngü Deyimleri

## BS-515 Nesneye Yönelik Programlama

Ders #3 (4 Kasım 2009)

# İçerik

- Geçen ders:
  - “Nesne” ve “Sınıf” Kavramları
  - “Nesne” ve “Sınıf”ın Java’da gerçekleştirilmesi

- Bu ders:
  - Java’da işleçler (“operators”)
  - Java’da kontrol-akış (“control-flow”) ve döngü (“loop”) deyimleri
  - Java konsol programlarında basit girdi/çıktı (“input/output”)

# Java’da İşleçler (“Operators”)

# İşleçler

- İşleçler, bir veya daha çok işlenen üzerinde işlem yapar.
  - Tek işlenen → “Unary operator”
  - İki işlenen → “Binary operator”

- “Unary operator”
  - <operator> <operand>  (prefix, ++i)
  - <operand> <operator>  (postfix, i++)

- “Binary operator”
  - <operand1> <operator> <operand2>  (infix, a+b)

# Aritmetik İşleçler - 1

| İşleç | Kullanım | Tanım |
|---|---|---|
| + | op1 + op2 | op1 ve op2'yi ekler |
| - | op1 - op2 | op2'yi op1'den çıkarır |
| * | op1 * op2 | op1 ve op2'yi çarpar |
| / | op1 / op2 | op1'i op2'ye böler |
| % | op1 % op2 | op1'in op2'ye göre modunu (bölme sonucunda kalanı) hesaplar |

# Aritmetik İşleçler - 2

- Java'da "+" işleci aynı zamanda dizgi ("String") birleştirmek için kullanılır.
  - System.out.println ("Girdinin" + sayi + "karakteri var.");
  - Çıktı: "Girdinin 5 karakteri var."

- "+" ve "-" işleçleri ayrıca işlenenin işaretini belirlemek için kullanılır.

| İşleç | Kullanım | Tanım |
|---|---|---|
| + | +op | İşlenen pozitif bir değer |
| - | -op | İşlenen negatif bir değer |

# Aritmetik İşleçler - 3

- “++” ve “--” işleçleri, bir değişkenin değerini arttırmak veya azaltmak için kullanılır.

| İşleç | Kullanım | Tanım |
|---|---|---|
| ++ | op++ | op değerini 1 arttırır. |
| -- | op-- | op değerini 1 azaltır. |

# İlişkisel ("Relational") İşleçler

- İki değeri karşılaştırarak aralarındaki ilişkiyi belirler.

| İşleç | Kullanım | Koşul sağlandığında 1 döndürür |
|---|---|---|
| > | op1>op2 | op1, op2'den büyükse |
| >= | op1>=op2 | op1, op2'den büyük veya op2'ye eşitse |
| < | op1<op2 | op1, op2'den küçükse |
| <= | op1<=op2 | op1, op2'den küçük veya op2'ye eşitse |
| == | op1==op2 | op1 ve op2 eşitse |
| != | op1!=op2 | op1 ve op2 eşit değilse |

# Şartlı ("Conditional") İşleçler

| İşleç | Kullanım | Koşul sağlandığında 1 döndürür |
|---|---|---|
| && | op1 && op2 | op1 ve op2'nin her biri doğruysa |
| \|\| | op1 \|\| op2 | op1 veya op2'den biri doğruysa |
| ! | ! op | op yanlışsa |

- *((sayi > GIRDI_SAYISI) && (System.in.read() !=-1))*
  - Önce soldaki kısım değerlendirilir.
  - Sol kısım doğruysa sağdaki kısım değerlendirilir; yanlışsa sağdaki kısım değerlendirilmez.

- *((sayi > GIRDI_SAYISI) || (System.in.read() !=-1))*
  - Her iki kısım da ayrı ayrı değerlendirilir.

# Bit (“Bitwise”) İşleçleri

- Verinin bitleri üzerinde işlem yaparlar.

| İşleç | Kullanım | Tanım |
|---|---|---|
| >> | op1 >> op2 | op1’in bitlerini op2 kadar sağa kaydırır. |
| << | op1 << op2 | op1’in bitlerini op2 kadar sola kaydırır |
| >>> | op1 >>> op2 | op1’in bitlerini op2 kadar sağa kaydırır (işaretsiz sayılar). |
| & | op1 & op2 | bit bazında “and” işlemi |
| \| | op1 \| op2 | bit bazında “or” işlemi |
| ^ | op1 ^ op2 | bit bazında “xor” işlemi |
| ~ | op1 ~ op2 | bit bazında tümleyen (“complement”) |

# Atama ("Assignment") İşleçleri

| İşleç | Kullanım | Tanım |
|---|---|---|
| += | op1 += op2 | op1 = op1 + op2 |
| -= | op1 -= op2 | op1 = op1 - op2 |
| *= | op1 *= op2 | op1 = op1 * op2 |
| /= | op1 /= op2 | op1 = op1 / op2 |
| %= | op1 %= op2 | op1 = op1 % op2 |
| &= | op1 &= op2 | op1 = op1 & op2 |
| \|= | op1 \|= op2 | op1 = op1 \| op2 |
| ^= | op1 ^= op2 | op1 = op1 ^ op2 |
| <<= | op1 <<= op2 | op1 = op1 << op2 |
| >>= | op1 >>= op2 | op1 = op1 >> op2 |
| >>>= | op1 >>>= op2 | op1 = op1 >>> op2 |

# Koşul (“Conditional”) İşleçleri

- ? :

- Örnek:

```
if (a > b) {
    max = a;
}
else {
    max = b;
}
```

```
max = (a > b) ? a : b;
```

# Java'da İşleç Önceliği

| | |
|---|---|
| postfix işleçler | [] . (parametreler) ifade++ ifade-- |
| birli (unary) işleçler | ++ ifade -- ifade + ifade - ifade ! |
| nesne oluşturma | new (tip) ifade |
| toplama/çıkarma | + - |
| kaydırma | << >> >>> |
| ilişkisel | < > <= >= instanceof |
| eşitlik | == != |
| bit AND | & |
| bit XOR | ^ |
| bit OR | \| |
| mantıksal AND | && |
| mantıksal OR | \|\| |
| koşul | ? : |
| atama | = |

# Java’da Kontrol-Akış (“Control-Flow”) ve Döngü (“Loop”) Deyimleri

# Kontrol-Akış Deyimleri

| Deyim | Anahtar Sözcük |
|---|---|
| Karar verme | if - else, switch - case |
| Döngü | for, while, do - while |
| Kural-dışı durum | try - catch - finally, throw |
| Diğer | break, continue, label:, return |

# “If … Else …” Deyimi

- Koşula bağlı olarak dallanma sağlar.
- Sözdizimi: If (*koşul*) *ifade-1* // koşul sağlanırsa ifade-1 işletilir.
  else *ifade-2* // koşul sağlanmazsa ifade-2 işletilir.
- Örnek:

```
…
// kullanıcı tarafından basılan “tamam” veya “iptal”
// düğmesine göre işlem yapılır.
…
if (yanıt == OK) {              // koşul: yanıt “tamam” mı?
                …
                // “tamam”a basılması halinde yapılacak işler
                …
                }
else {
                …
                // “iptal”e basılması halinde yapılacak işler
                …
                }
```

# “If … ” Deyimi: Örnek

```
class Merhaba {
    public static void main (String args[]) {
        if (args.length > 0) {
            System.out.println(“Merhaba " + args[0]);
            }
        }
    }
```

# “If … Else …” Deyimi: Örnek

```
class Merhaba {
    public static void main (String args[]) {
        if (args.length > 0) {
                System.out.println("Merhaba " + args[0]);
                }
        else {
                System.out.println(“Her kimsen merhaba!");
                }
        }
    }
```

# “If … Else If …” Deyimi: Örnek

```
class Merhaba {
    public static void main (String args[]) {
        if (args.length == 0) {
            System.out.println(“Her kimsen merhaba!");
        } else if (args.length == 1) {
            System.out.println(“Merhaba " + args[0]);
        } else if (args.length == 2) {
            System.out.println("Merhaba " + args[0] + " " + args[1]);
        } else if (args.length == 3) {
            System.out.println("Merhaba " + args[0] + " " + args[1] + " “ + args[2]);
        } else if (args.length == 4) {
            System.out.println("Merhaba " + args[0] + " " + args[1] + " " + args[2] + " " + args[3]);
        } else {
            System.out.println("Merhaba " + args[0] + " " + args[1] + " " + args[2] + " " + args[3] + " ve diğerleri!");
        }
    }
}
```

# Eşleşmemiş “Else” Problemi

```
if (x>5)
     if (y>5)
         System.out.println("x ve y 5 den buyuk");
else
     System.out.println("x 5 den kucuk ya da esit");
```

YANLIŞ !

```
if (x>5) {
     if (y>5)
         System.out.println("x ve y 5 den buyuk");
}
else
     System.out.println("x 5 den kucuk ya da esit");
```

DOĞRU !

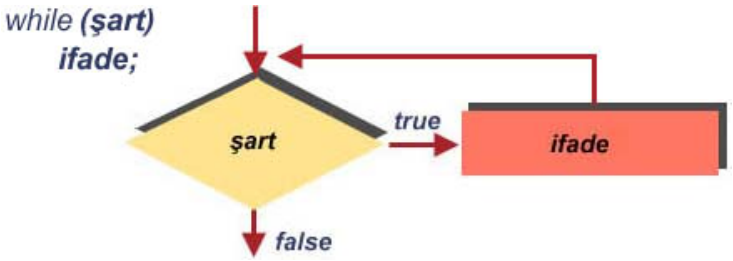
# “Switch” Deyimi

- Birden çok seçeneği olan koşulda, her seçenek için ifadeleri işletmek amacıyla kullanılır.
- Örnek:

```
int mevsim;
…
switch (mevsim) {
case  1: System.out.println (“Kış”); break;
case  2:  System.out.println (“Bahar”); break;
case  3:  System.out.println (“Yaz”); break;
case  4:  System.out.println (“Sonbahar”); break;
default: System.out.println (“Yanlış mevsim!”); break;
}
```

# Döngü Deyimleri

- while (*koşul*) {
  *ifadeler*
  }



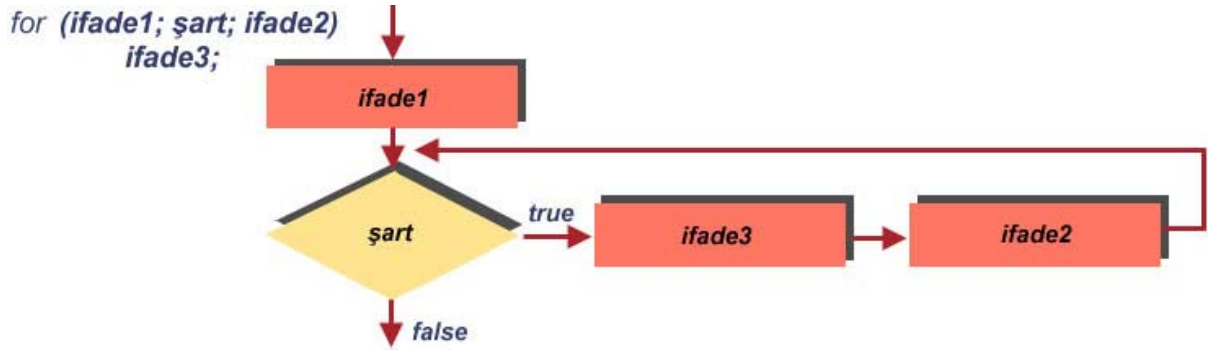

- for (*döngü sayacını ilklendirme*; *döngü sonlandırma koşulu*; *döngü sayacını arttırma*) {
  *ifadeler*
  }

- do {
  *ifadeler*
  } while (*booleİfade*);

# “While” Döngüsü

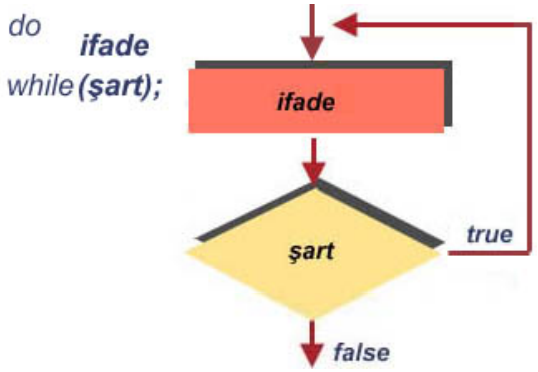
```
class Merhaba {
    public static void main (String args[]) {
        int i;
        System.out.print ("Merhaba ");          // “Merhaba ” yaz
        i = 0;                                  // döngü sayacını ilklendir
        while (i < args.length) {               // koşulu test et; doğruysa döngüye gir
            System.out.print (args[i]);         // parametreyi yaz
            System.out.print(" ");              // boşluk yaz
            i = i + 1;                          // döngü sayacını arttır
            }
        System.out.println();                   // satırı bitir
        }
    }
```

# “For” Döngüsü




```
class Merhaba {
    public static void main (String args[]) {
        System.out.print("Merhaba ");                    // “Merhaba ” yaz
        for (int i = 0; i < args.length; i = i + 1) {    // koşulu test et; doğruysa döngüye gir
            System.out.print(args[i]);                   // değiştirgeyi yaz
            System.out.print(" ");    // boşluk yaz
            }
        System.out.println();                            // satırı bitir
        }
    }
```

# “Do While” Döngüsü




```
class Merhaba {
    public static void main (String args[]) {
        int i = -1;
        do {
            if (i == -1)                          // koşul doğruysa
                System.out.print("Merhaba ");     // “Merhaba ” yaz
            else {
                System.out.print(args[i]);        // koşul doğru değilse
                System.out.print(" ");            // boşluk yaz
            }
            i = i + 1;                            // döngü sayacını arttır
        } while (i < args.length);                // hala yazacak değiştirge varsa devam et
        System.out.println();                     // satırı bitir
    }
}
```

# “Break” Deyimi

- “Break” deyimi, giriş koşulunun geçersiz olmasını beklemeden döngüden çıkmayı sağlar.

```
class Say {
        public static void main (String args[]) {
        int toplam = 0;
        int sayi = 1;
                for (int kare = 1; kare <= 64; kare++) {
                        sayi = sayi * 2;
                        if (sayi <= 0) {
                                System.out.println("Hata: Taşma Oluştu!");
                                break;
                        }
                        toplam = toplam + sayi;
                        System.out.print(toplam + "\t ");
                }
        }
}
```

# “Continue” Deyimi

- “Continue” deyimi, kalan komutların işletilmesine izin vermeden, program akışını en içteki döngünün başına geçirir.

- Örnek:

```
for (int i = 0; i < m.length; i++) {
        if (m[i] % 2 == 0) continue;
                // çift ise döngüye bir sonraki döngüye geç

        // tek ise devam eden komutları işleme al
        }
```

# Dallanma Deyimleri

- Etiketli “break”

- Örnek:

```
test:  if (check(i)) {
           for (int j=0; j<10; j++) {
               if (j>i) break;              // sadece bu döngüyü sonlandır.
               if (a[i][j] == null)
                   break test;    // if bloğunu sonlandır.
               }
           }
```

# Örnek

- Bir sınıftaki öğrencilerin o dersin sınavından aldıkları notların ortalamalarını bulan konsol programını yazalım
  - Kontrol değeri (“sentinel”) kullanarak
    - “while”, “do … while” döngüleri ile
  - Sayaç değeri (“loop counter”) kullanarak
    - “while”, “for” döngüleri ile

# Java’da Basit Girdi/Çıktı (“Input/Output”)

# Java Konsol Programlarının Yapısı

- Her Java konsol programı, bir ya da birden fazla yordamı içeren bir sınıftan oluşur ve bu yordamlardan biri ana yordam (“main method”) olmalıdır.

```
imported classes
public class  sınıfın-adı   {
      public static void main (String args[]) throws IOException {
            değişkenlerin tanımlanması
            çalıştırılabilir ifadeler
       }
       varsa diğer yordamlar
}
```

Çalıştırma:

> javac MerhabaDunya.java ⇨ MerhabaDunya.class
> java MerhabaDunya
> Merhaba Dünya!

# Konsol Programlarında Basit Çıktı

- System.out.println("bir dizgi");
  - > Bir dizgi
- System.out.println("bir dizgi");
  System.out.println("diger dizgi");
  - > bir dizgi
  - > diğer dizgi
- System.out.print("bir dizgi, ");
  System.out.print("diğer dizgi");
  - > bir dizgi, diğer dizgi

# Konsol Programlarında Basit Girdi

- BufferedReader stdGirdi =
  new BufferedReader (new InputStreamReader(System.in));

- Dizgi okuma:
  - String s = stdGirdi.readLine();

- Tamsayı okuma:
  - String s = stdGirdi.readLine();
    int i = new Integer(s.trim()).intValue();
    // ya da;
    int j = new Integer.parseInt(s.trim());

- Reel sayı okuma:
  - String s = stdGirdi.readLine();
    double d = new Double(s.trim()).doubleValue();

# Örnek

- İki tamsayıyı klavyeden okuyan ve bu iki sayının toplamını ve birinci ile ikinci sayı arasındaki farkı bularak ekrana bastıran Java konsol programı
  - ToplaCikar.java